Activity 1 - Blink the internal LED (pin13)

This program is a simple set-up to make the internal LED (pin 13) blink at 1 second intervals.

Circuit set-up: Simply connect the Uno to the USB port.

Code:
This program is in the examples folder of the Arduino IDE. Go to Files – Examples _ Basics and select Blink. Make sure that you have selected the correct board (Uno) and port. Upload the Sketch

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Comments
The internal LED will link at one second intervals. Can you make it blink at ½ second intervals? How about 2 second intervals?

Activity 2 – Blink an external LED

In this activity you will connect an external LED to the Arduino and make it blink. The LED will only work in on direction and will need a resistor to ensure it works correctly. Looking at the LED the longer pin should be connected to the pin and the shorter pin to a resistor and then to ground.

Circuit Set-up: See Instruction sheet: Activity 2

Code:

The code is identical to that used in Activity 1. This is because the same pin is being used to control the LED.

Comment:
Can you use a different pin to control the LED? Can you vary the blink rate? How about coding for a second LED?

Activity 3 – Controlling an LED with a button

Having the LED work all the time is one thing, but how do we make it respond to an input, such as a button press?

Circuit Set-up: See Instruction sheet – Activity 3

Code:

```
/*
  Button

 Turns on and off a light emitting diode(LED) connected to digital
 pin 13, when pressing a pushbutton attached to pin 2.


 The circuit:
 * LED attached from pin 13 to ground
 * pushbutton attached to pin 11 from +5V
 * resistor attached to pin 2 from ground
*/

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 11;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Comment:

Now when we press the button we can control the output of the input and this is read by the LED. When the button is pressed (the input is high) the LED will light. Can you think how to program the button to turn the LED off if it is on?

Activity 4: Controlling an LED using pulse width modulation(pwm)

Digital pins normally have only 2 states, on or off (high or low). Sometimes however we might want to vary the output. This can be done using pulse width modulation (pwm). The digital pin is given a duty cycle (power is only given for a percentage of the time). This means that if a duty cycle is 50% the pin will only appear to have 50% of the energy. Hence it is possible to vary the output of a digital pin.

Circuit Set-up: See Instruction sheet - Activity 4

Code:

```
/*
 Fade

 This example shows how to fade an LED on pin 9
 using the analogWrite() function.
On most  Arduino’s, the PWM pins are identified with
 a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

 */

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
```

```
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Comment:

Try changing the fade amount. Can you make a second LED fade on a different pin? (Check which pins work – look at the code!)

Activity 5: Controlling an LED with a potentiometer (variable resistor)

Although we can change control the output using pwm, sometimes we might want to use a potentiometer to move to a particular output. In this sketch you will use a potentiometer to control the speed of a blinking LED.

Circuit Set-up: See Instruction sheet – Activity 5

Code:

```
/*Potentiometer LED Sketch
*/

const int potPin = 0;
const int ledPin = 13;
int val = 0;

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  val = analogRead(potPin);
  digitalWrite(ledPin, HIGH);
  delay(val);
  digitalWrite(ledPin, LOW);
  delay(val);

}
```

Comment:

You can use the same idea to vary the output brightness of an LED or speed of a motor in the same way.

Activity 6: Rainbow colours (RGB LED)

Most LEDs only emit one colour, although we can alter the brightness. An RGB LED actually has three different colours which can be controlled to produce specific colours of patterns.

Circuit Set-up: See Instruction sheet – Activity 6

Code:

```
const int redPin = 11;  // R petal on RGB LED module connected to digital pin 11
const int greenPin = 10;  // G petal on RGB LED module connected to digital pin 10
const int bluePin = 9;  // B petal on RGB LED module connected to digital pin 9

void setup()
{
	pinMode(redPin, OUTPUT); // sets the redPin to be an output
	pinMode(greenPin, OUTPUT); // sets the greenPin to be an output
	pinMode(bluePin, OUTPUT); // sets the bluePin to be an output
}

void loop()  // run over and over again
{
	// Basic colors:
	color(255, 0, 0); // turn the RGB LED red
	delay(1000); // delay for 1 second
	color(0,255, 0); // turn the RGB LED green
	delay(1000); // delay for 1 second
	color(0, 0, 255); // turn the RGB LED blue
	delay(1000); // delay for 1 second
	// Example blended colors:
	color(255,0,0); // turn the RGB LED red
	delay(1000); // delay for 1 second
	color(237,109,0); // turn the RGB LED orange
	delay(1000); // delay for 1 second
	color(255,215,0); // turn the RGB LED yellow
	delay(1000); // delay for 1 second
	color(0,255,0); // turn the RGB LED green
	delay(1000); // delay for 1 second
	color(0,0,255); // turn the RGB LED blue
	delay(1000); // delay for 1 second
	color(0,46,90); // turn the RGB LED  indigo
	delay(1000); // delay for 1 second
	color(128,0,128); // turn the RGB LED purple
	delay(1000); // delay for 1 second
}
```

```
void color (unsigned char red, unsigned char green, unsigned char blue)     // the color generating
function
{
        analogWrite(redPin, red);
        analogWrite(bluePin, blue);
        analogWrite(greenPin, green);
}
```

Comment:

Try experimenting by changing the colours (numbers). The maximum for each LED is 255 and they are written in the order Red, Blue, Green in the code.

Activity 7: Serial Monitor

In this activity you will use the serial port to send a signal to control which colour LED is lit.

Circuit Set-up: See Instruction sheet – Activity 7

Code:

```
//Serial Monitor
// open the serial monitor, if you input red, you will see the red LED light up
const int greenPin= 2; //the green led pin attact to
const int yellowPin= 3; //the yellow led pin attact to
const int redPin= 4; //the red led pin attach to
String comdata = "";
int lastLength = 0;

void setup()
{
  pinMode(greenPin,OUTPUT);  //initialize the greenPin as output
  pinMode(yellowPin, OUTPUT);  //initialize the yellowPin as output
  pinMode(redPin, OUTPUT);  //initialize the redPin as output
  Serial.begin(9600);  // start serial port at 9600 bps:
  Serial.print("Please input any colour of LED:");  //print message on serial monitor
}

void loop()
{
  //read string from serial monitor
  if(Serial.available()>0)  // if we get a valid byte, read analog ins:
  {
    comdata = "";
    while (Serial.available() > 0)
    {
      comdata += char(Serial.read());
      delay(2);
    }
    Serial.println(comdata);
```

```
 }
  if(comdata == "red")
  {
    digitalWrite(redPin, HIGH);//turn the red led on
    digitalWrite(greenPin, LOW);//turn the green led off
    digitalWrite(yellowPin, LOW);//turn the yellow led off
  }
  else if(comdata == "yellow")
  {
    digitalWrite(redPin, LOW);//turn the red led off
    digitalWrite(greenPin, LOW);//turn the green led off
    digitalWrite(yellowPin, HIGH);//turn the yellow led on
  }
  else if(comdata == "green")
  {
    digitalWrite(redPin, LOW);//turn the red led off
    digitalWrite(greenPin, HIGH);//turn the green led on
    digitalWrite(yellowPin, LOW);//turn the yellow led off
  }
  else
  {
    digitalWrite(redPin, LOW);//turn the red led off
    digitalWrite(greenPin, LOW);//turn the green led off
    digitalWrite(yellowPin, LOW);//turn the yellow led off
  }
}
```

Comment:

To run this program you will need to open the serial monitor (upper right corner of the IDE). You should then see a window with the words “Please input any colour of LED”. If you type yellow, green or red, the corresponding LED will light up. Any other input – nothing will happen.

It is also possible to receive data from the Arduino which is how most sensors work.

Activity 8: LCD Welcome screen

LCD (liquid crystal displays) are very common for presenting simple information and instructions. In this sketch you will create a simple Welcome message.

Circuit Set-up: See Instruction sheet – Activity 8

Code:

```
 /* name:I2C LCD1602
  * function:You should now see your I2C LCD1602 display the flowing characters: "SunFounder" and
"hello, world".
  */
```

```
// include the library code
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

char array1[]=" Awesome              ";  //the string to print on the LCD
char array2[]="hello, world!            ";  //the string to print on the LCD
int tim = 500;  //the value of delay time
// initialize the library with the numbers of the interface pins
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line display

void setup()
{
 lcd.init();  //initialize the lcd
 lcd.backlight();  //open the backlight
}

void loop()
{
 lcd.setCursor(15,0);  // set the cursor to column 15, line 0
 for (int positionCounter1 = 0; positionCounter1 < 26; positionCounter1++)
 {
  lcd.scrollDisplayLeft();  //Scrolls the contents of the display one space to the left.
  lcd.print(array1[positionCounter1]);  // Print a message to the LCD.
  delay(tim);  //wait for 250 microseconds
 }
 lcd.clear();  //Clears the LCD screen and positions the cursor in the upper-left corner.
 lcd.setCursor(15,1);  // set the cursor to column 15, line 1
 for (int positionCounter = 0; positionCounter < 26; positionCounter++)
 {
  lcd.scrollDisplayLeft();  //Scrolls the contents of the display one space to the left.
  lcd.print(array2[positionCounter]);  // Print a message to the LCD.
  delay(tim);  //wait for 250 microseconds
 }
 lcd.clear();  //Clears the LCD screen and positions the cursor in the upper-left corner.
}
```

Comment:

Now you can try experimenting with different messages and scrolling. You can connect LCD screens to allow input, via keypads and to give information about sensors.

Activity 9: Voltmeter

Here you will use the LCD to display the voltage provided by a potentiometer.

Circuit Set-up: See Instruction sheet – Activity 9

Code:

```
// include the library code
```

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line display
float val = 0;

void setup()
{
  Serial.begin(9600);//initialize the serial
  lcd.init();  //initialize the lcd
  lcd.backlight();  //open the backlight
  lcd.print("Voltage Value:");//print "Voltage Value:"
}

void loop()
{
  val = analogRead(A0);//Read the value of the potentiometer to val
  val = val/1024*5.0;// Convert the data to the corresponding voltage value in a math way
  Serial.print(val);//Print the number of val on the serial monitor
  Serial.print("V"); // print the unit as V, short for voltage on the serial monitor
   Serial.println( );
  lcd.setCursor(6,1);//Place the cursor at Line 1, Column 6. From here the characters are to be
displayed
  lcd.print(val);//Print the number of val on the LCD
  lcd.print("V");//Then print the unit as V, short for voltage on the LCD
  delay(200); //Wait for 200ms
}
```

Comment:

The potentiometer 'splits' the voltage and this is displayed on the LCD screen. This is in essence how many sensors can display data.

Activity 10: 7-Segment LED

Ideal for displaying numbers and letters, and the introduction to more complex displays. This program is longer as you need to define each of the numbers and letters to display.

Circuit Set-up: See Instruction sheet – Activity 10

Code:

```
//7-Segment Display
//You should now see the 7-segment display cycle from 0 to F

const int a=7; //a of 7-segment attach to digital pin 7
const int b=6; //b of 7-segment attach to digital pin 6
const int c=5; //c of 7-segment attach to digital pin 5
const int d=11;//d of 7-segment attach to digital pin 11
const int e=10;//e of 7-segment attach to digital pin 10
const int f=8;//f of 7-segment attach to digital pin 8
const int g=9;//g of 7-segment attach to digital pin 9
const int dp=4;//dp of 7-segment attach to digital pin 4

void setup()
{
//loop over thisPin from 4 to 11 and set them all to output
for(int thisPin = 4;thisPin <= 11;thisPin++)
{
pinMode(thisPin,OUTPUT);
}
digitalWrite(dp,LOW);//turn the dp of the 7-segment off
}

void loop()
{
digital_1();//display 1 to the 7-segment
delay(1000);//wait for a second
digital_2();//display 2 to the 7-segment
delay(1000); //wait for a second
digital_3();//display 3 to the 7-segment
delay(1000); //wait for a second
digital_4();//display 4 to the 7-segment
delay(1000); //wait for a second
digital_5();//display 5 to the 7-segment
delay(1000); //wait for a second
digital_6();//display 6 to the 7-segment
delay(1000); //wait for a second
digital_7();//display 7 to the 7-segment
delay(1000); //wait for a second
digital_8();//display 8 to the 7-segment
delay(1000); //wait for a second
digital_9();//display 8 to the 7-segment
delay(1000); //wait for a second
digital_A();//display 8 to the 7-segment
delay(1000); //wait for a second
digital_b();//display 8 to the 7-segment
delay(1000); //wait for a second
digital_C();//display 8 to the 7-segment
delay(1000); //wait for a second
digital_d();//display 8 to the 7-segment
delay(1000); //wait for a second
digital_E();//display 8 to the 7-segment
delay(1000); //wait for a second
digital_F();//display 8 to the 7-segment
delay(1000); //wait for a second
}

void digital_1(void) //display 1 to the 7-segment
{
digitalWrite(c,HIGH);//turn the c of the 7-segment on
digitalWrite(b,HIGH);//turn the b of the 7-segment on
for(int j = 7;j <= 11;j++)//turn off the others
digitalWrite(j,LOW);
}
void digital_2(void) //display 2 to the 7-segment
{
digitalWrite(b,HIGH);
digitalWrite(a,HIGH);
for(int j = 9;j <= 11;j++)
digitalWrite(j,HIGH);
digitalWrite(c,LOW);
digitalWrite(f,LOW);
}
void digital_3(void) //display 3 to the 7-segment
{
unsigned char j;
digitalWrite(g,HIGH);
digitalWrite(d,HIGH);
for(j=5;j<=7;j++)
digitalWrite(j,HIGH);
digitalWrite(f,LOW);
digitalWrite(e,LOW);
}
void digital_4(void) //display 4 to the 7-segment
{
digitalWrite(c,HIGH);
digitalWrite(b,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(a,LOW);
digitalWrite(e,LOW);
digitalWrite(d,LOW);
}
void digital_5(void) //display 5 to the 7-segment
{
unsigned char j;
for(j = 7;j <= 9;j++)
digitalWrite(j,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(b,LOW);
digitalWrite(e,LOW);
}
void digital_6(void) //display 6 to the 7-segment
{
unsigned char j;
for(j = 7;j <= 11;j++)
digitalWrite(j,HIGH);
digitalWrite(c,HIGH);
digitalWrite(b,LOW);
}
void digital_7(void) //display 7 to the 7-segment
```

```
{
unsigned char j;
for(j = 5;j <= 7;j++)
digitalWrite(j,HIGH);
for(j = 8;j <= 11;j++)
digitalWrite(j,LOW);
}
void digital_8(void) //display 8 to the 7-segment
{
unsigned char j;
for(j = 5;j <=11;j++)
digitalWrite(j,HIGH);
}
void digital_9(void) //display 9 to the 7-segment
{
unsigned char j;
for(j = 5;j <=9;j++)
digitalWrite(j,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
}
void digital_A(void) //display A to the 7-segment
{
unsigned char j;
for(j = 5;j <=10;j++)
digitalWrite(j,HIGH);
digitalWrite(d,LOW);
}
void digital_b(void) //display b to the 7-segment
{
unsigned char j;
for(j = 7;j <=11;j++)
digitalWrite(j,HIGH);
digitalWrite(a,LOW);
digitalWrite(b,LOW);
}
void digital_C(void) //display C to the 7-segment
{
digitalWrite(a,HIGH);
digitalWrite(b,LOW);
digitalWrite(c,LOW);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,LOW);
}
void digital_d(void) //display d to the 7-segment
{
unsigned char j;
digitalWrite(a,LOW);
digitalWrite(f,LOW);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(j,HIGH);
for(j = 9;j <=11;j++)
digitalWrite(j,HIGH);
}
void digital_E(void) //display E to the 7-segment
{
unsigned char j;
digitalWrite(b,LOW);
digitalWrite(c,LOW);
for(j = 7;j <=11;j++)
digitalWrite(j,HIGH);
}
void digital_F(void) //display F to the 7-segment
{
unsigned char j;
digitalWrite(b,LOW);
digitalWrite(c,LOW);
digitalWrite(d,LOW);
for(j = 7;j <=10;j++)
digitalWrite(j,HIGH);
}
```

Comment:

Now that you know the code for each number and letter you can create your own patterns and sequences.

Activity 11: 4-digit 7-segment LED (Stopwatch)

Following on from the last activity, you will now build a 4-digit screen capable of showing time and counting down.

Circuit Set-up: See Instruction sheet – Activity 11

Code:

```
#include <TimerOne.h>
//the pins of 4-digit 7-segment display attach to pin2-13 respectively
int a = 2;
int b = 3;
int c = 4;
int d = 5;
int e = 6;
int f = 7;
int g = 8;
int p = 9;

int d4 = 10;
int d3 = 11;
int d2 = 12;
int d1 = 13;

long n = 0;// n represents the value displayed on the LED display. For example, when n=0, 0000 is
displayed. The maximum value is 9999.
int x = 100;
int del = 5;//Set del as 5; the value is the degree of fine tuning for the clock
int count = 0;//Set count=0. Here count is a count value that increases by 1 every 0.1 second, which
means 1 second is counted when the value is 10

void setup()
{
 //set all the pins of the LED display as output
 pinMode(d1, OUTPUT);
 pinMode(d2, OUTPUT);
 pinMode(d3, OUTPUT);
 pinMode(d4, OUTPUT);
 pinMode(a, OUTPUT);
 pinMode(b, OUTPUT);
 pinMode(c, OUTPUT);
 pinMode(d, OUTPUT);
 pinMode(e, OUTPUT);
 pinMode(f, OUTPUT);
 pinMode(g, OUTPUT);
 pinMode(p, OUTPUT);

 Timer1.initialize(100000); // set a timer of length 100000 microseconds (or 0.1 sec - or 10Hz => the
led will blink 5 times, 5 cycles of on-and-off, per second)
 Timer1.attachInterrupt( add ); // attach the service routine here
}

void loop()
{
 clearLEDs();//clear the 7-segment display screen
 pickDigit(0);//Light up 7-segment display d1
 pickNumber((n/1000));// get the value of thousand
 delay(del);//delay 5ms

 clearLEDs();//clear the  7-segment display screen
 pickDigit(1);//Light up 7-segment display d2
 pickNumber((n%1000)/100);// get the value of hundred
 delay(del);//delay 5ms

 clearLEDs();//clear the  7-segment display screen
 pickDigit(2);//Light up 7-segment display d3
 pickNumber(n%100/10);//get the value of ten
 delay(del);//delay 5ms

 clearLEDs();//clear the 7-segment display screen
 pickDigit(3);//Light up 7-segment display d4
 pickNumber(n%10);//Get the value of single digit
 delay(del);//delay 5ms
}

void pickDigit(int x)  //light up a 7-segment display
{
 //The 7-segment LED display is a common-cathode one. So also use digitalWrite to set d1 as high
and the LED will go out
 digitalWrite(d1, HIGH);
 digitalWrite(d2, HIGH);
 digitalWrite(d3, HIGH);
 digitalWrite(d4, HIGH);

 switch(x)
 {
  case 0:
  digitalWrite(d1, LOW);//Light d1 up
  break;
  case 1:
  digitalWrite(d2, LOW); //Light d2 up
  break;
  case 2:
  digitalWrite(d3, LOW); //Light d3 up
  break;
 default:
  digitalWrite(d4, LOW); //Light d4 up
  break;
 }
}
//The function is to control the 7-segment LED display to display numbers. Here x is the number to be
displayed. It is an integer from 0 to 9
void pickNumber(int x)
{
 switch(x)
 {
  default:
  zero();
  break;
  case 1:
  one();
  break;
  case 2:
  two();
  break;
  case 3:
  three();
  break;
  case 4:
  four();
  break;
 case 5:
  five();
  break;
 case 6:
  six();
  break;
 case 7:
  seven();
  break;
 case 8:
  eight();
  break;
 case 9:
  nine();
  break;
 }
}
void clearLEDs() //clear the  7-segment display screen
{
 digitalWrite(a, LOW);
 digitalWrite(b, LOW);
 digitalWrite(c, LOW);
 digitalWrite(d, LOW);
 digitalWrite(e, LOW);
 digitalWrite(f, LOW);
 digitalWrite(g, LOW);
 digitalWrite(p, LOW);
}

void zero() //the  7-segment led display 0
{
 digitalWrite(a, HIGH);
 digitalWrite(b, HIGH);
 digitalWrite(c, HIGH);
 digitalWrite(d, HIGH);
 digitalWrite(e, HIGH);
 digitalWrite(f, HIGH);
 digitalWrite(g, LOW);
}

void one()  //the  7-segment led display 1
{
 digitalWrite(a, LOW);
 digitalWrite(b, HIGH);
 digitalWrite(c, HIGH);
 digitalWrite(d, LOW);
 digitalWrite(e, LOW);
 digitalWrite(f, LOW);
 digitalWrite(g, LOW);
}

void two()  //the  7-segment led display 2
{
 digitalWrite(a, HIGH);
 digitalWrite(b, HIGH);
 digitalWrite(c, LOW);
 digitalWrite(d, HIGH);
 digitalWrite(e, HIGH);
 digitalWrite(f, LOW);
 digitalWrite(g, HIGH);
}

void three()  //the  7-segment led display 3
{
 digitalWrite(a, HIGH);
 digitalWrite(b, HIGH);
 digitalWrite(c, HIGH);
 digitalWrite(d, HIGH);
 digitalWrite(e, LOW);
 digitalWrite(f, LOW);
 digitalWrite(g, HIGH);
}

void four()  //the  7-segment led display 4
{
 digitalWrite(a, LOW);
 digitalWrite(b, HIGH);
 digitalWrite(c, HIGH);
 digitalWrite(d, LOW);
 digitalWrite(e, LOW);
 digitalWrite(f, HIGH);
 digitalWrite(g, HIGH);
}

void five()  //the  7-segment led display 5
{
 digitalWrite(a, HIGH);
 digitalWrite(b, LOW);
 digitalWrite(c, HIGH);
 digitalWrite(d, HIGH);
 digitalWrite(e, LOW);
 digitalWrite(f, HIGH);
 digitalWrite(g, HIGH);
}

void six()  //the  7-segment led display 6
{
 digitalWrite(a, HIGH);
 digitalWrite(b, LOW);
 digitalWrite(c, HIGH);
 digitalWrite(d, HIGH);
 digitalWrite(e, HIGH);
 digitalWrite(f, HIGH);
 digitalWrite(g, HIGH);
}

void seven()  //the  7-segment led display 7
```

```
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
}

void eight()   //the  7-segment led display 8
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}

void nine()  //the  7-segment led display 9
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}

void add()
{
    // Toggle LED
    count ++;
    if(count == 10)
    {
      count = 0;
      n ++;
      if(n == 10000)
      {
        n = 0;
      }
    }
}
```

Comment:

Again the code needs to define all the possible numbers and is quite long. (This is where being able to copy and paste can come in very useful – and save lots of time.)

Activity 12: Servo control

There are different types of motors that can be used for different purposes. The servo is particularly suited to robotics and projects as it can be controlled very precisely to move to defined positions.

Circuit Set-up: See Instruction sheet – Activity 12

Code:

```
#include <Servo.h>
Servo myservo;//create servo object to control a servo

void setup()
{
  myservo.attach(9);//attachs the servo on pin 9 to servo object
  myservo.write(0);//back to 0 degrees
  delay(1000);//wait for a second
}

void loop()
{
  myservo.write(15);//goes to 15 degrees
  delay(1000);//wait for a second
  myservo.write(30);//goes to 30 degrees
  delay(1000);//wait for a second.33
  myservo.write(45);//goes to 45 degrees
  delay(1000);//wait for a second.33
  myservo.write(60);//goes to 60 degrees
  delay(1000);//wait for a second.33
  myservo.write(75);//goes to 75 degrees
  delay(1000);//wait for a second.33
  myservo.write(90);//goes to 90 degrees
  delay(1000);//wait for a second
  myservo.write(75);//back to 75 degrees
  delay(1000);//wait for a second.33
  myservo.write(60);//back to 60 degrees
  delay(1000);//wait for a second.33
  myservo.write(45);//back to 45 degrees
  delay(1000);//wait for a second.33
  myservo.write(30);//back to 30 degrees
  delay(1000);//wait for a second.33
  myservo.write(15);//back to 15 degrees
  delay(1000);//wait for a second
  myservo.write(0);//back to 0 degrees
  delay(1000);//wait for a second
}
```

Comment:

Servos are very useful to control movement and are used extensively in robotics to allow articulated movement.

Activity 13: Keypad

Keypads are very useful for inputting numbers and are commonly used to as pin entry tools, where they can be programmed to accept individual pin numbers to allow access to systems.

Circuit Set-up: See Instruction sheet – Activity 13

Code:

```
/*
 * name:Password Lock
 * function: the I2C LCD1602 will display "Welcome!" after power on.
 * At this point, the indicator LED on the relay keeps off.
 * When you press "*" key, it will prompt "Input Your Code:". If you enter “123456” and press “#” key
to confirm,
 * the indicator LED on the relay module will light up. The I2C LCD1602 will display "Input Correctly"
"Please Come In".
 * Two seconds later, "Welcome!"
 */

#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


LiquidCrystal_I2C lcd(0x27,16,2);


#define relayPin 13  //relay attach to pin 13


const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the cymbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','4'},
  {'5','6','7','8'},
  {'9','A','B','C'},
  {'D','*','0','#'}
};
```

```
byte rowPins[ROWS] = {11, 10, 9, 8}; //connect to the row pinouts of the keypad
byte colPins[COLS] = { 7, 6, 5, 4}; //connect to the column pinouts of the keypad


int pos = 0;
char secretCode[6] = {'1', '2', '3', '4', '5', '6'};
char inputCode[6] = {'0', '0', '0', '0', '0', '0'};

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup()
{
 lcd.init();
 lcd.backlight();
 pinMode(relayPin, OUTPUT);
 //Serial.begin(9600);
 lcd.setCursor(0,0);
 lcd.print("    Welcome!    ");
 delay(2000);
}

void loop()
{
 readKey();
}

void readKey()
{
 int correct = 0;
 int i;
 char customKey = customKeypad.getKey();
 if (customKey)
 {
  switch(customKey)
  {
   case '*':
    pos = 0;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Input Your Code:");
   break;
   case '#':
    for(i = 0; i < 6; i++)
    {
     if(inputCode[i] == secretCode[i])
     {
      correct ++;
     }
    }
    if(correct == 6)
    {
     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("Input correctly!");
     lcd.setCursor(0, 1);
     lcd.print(" Please Come In ");
    // digitalWrite(relayPin, HIGH);
    // delay(2000);
    // lcd.clear();
    // lcd.setCursor(0,0);
    // lcd.print("    Welcome!    ");
    // digitalWrite(relayPin, LOW);
    }
    else
    {
     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("  Input Error!  ");
     lcd.setCursor(0, 1);
     lcd.print("  Please Again  ");
     digitalWrite(relayPin, LOW);
     delay(2000);
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("    Welcome!    ");
    }
   break;
   default:
    inputCode[pos] = customKey;
    lcd.setCursor(pos,1);
    lcd.print(inputCode[pos]);
    pos ++;
  }
 }
}
```

Comment:

How could you use this to move a servo to a different position?